汇编语言程序设计
Assembly Language Programming

晁平复

课程简介

- ✓ 本课程是计算机科学与技术、软件工程等专业本科生的专业选修课程
- ✓ 本课程是教育部-华为“智能基座”产教融合协同育人基地项目立项课程
- ✓ 本课程主要起到了“上承高级语言，下启机器系统”的桥梁作用
- ✓ 课程的总体目标是，深入理解计算机系统的工作原理，全面提升高级语言程序设计能力

课程简介

➢ 学习内容

1、学习并掌握X86系列CPU的32位汇编语言程序设计

2、学习了解华为鲲鹏ARM处理器汇编语言程序设计

➢ 学习评价方式

1、平时表现、作业

2、课程实验

3、期末考试

➢ 知识依赖：C语言、Linux

第1章 基础知识

1.1 CPU基本功能

1.2 汇编语言

1.3 数据的表示和存储

1.1 CPU基本功能

1.1.1 目标代码

1.1.2 CPU基本功能

1.1.1 目标代码

✓计算机系统中的CPU只能执行机器指令

✓由机器指令组成的程序，被称为**目标程序**，也被称为**目标代码**

✓计算机系统最终运行的是目标程序

```
0000000000000000 <main>:
   0:   f3 0f 1e fa             endbr64
   4:   55                      push   %rbp
   5:   48 89 e5                mov    %rsp,%rbp
   8:   48 83 ec 20             sub    $0x20,%rsp
   c:   89 7d ec                mov    %edi,-0x14(%rbp)
   f:   48 89 75 e0             mov    %rsi,-0x20(%rbp)
  13:   83 7d ec 04             cmpl   $0x4,-0x14(%rbp)
  17:   74 16                   je     2f <main+0x2f>
  19:   48 8d 3d 00 00 00 00    lea    0x0(%rip),%rdi
                        1c: R_X86_64_PC32       .rodata-0x4
  20:   e8 00 00 00 00          callq  25 <main+0x25>
                        21: R_X86_64_PLT32      puts-0x4
  25:   bf 01 00 00 00          mov    $0x1,%edi
  2a:   e8 00 00 00 00          callq  2f <main+0x2f>
                        2b: R_X86_64_PLT32      exit-0x4
  2f:   c7 45 fc 00 00 00 00    movl   $0x0,-0x4(%rbp)
  36:   eb 48                   jmp    80 <main+0x80>
  38:   48 8b 45 e0             mov    -0x20(%rbp),%rax
  3c:   48 83 c0 10             add    $0x10,%rax
  40:   48 8b 10                mov    (%rax),%rdx
```

1.1.1 目标代码

计算1到10的平方之和

```
int  cf11( void )
{
    int   sum,  i;
    sum = 0;
    for ( i = 1;  i <= 10;  i += 1 )
       sum += i*i;
    return  sum;
}
```

1.1.1 目标代码

计算1到10的平方之和

```
int  cf11( void )
{
    int   sum,  i;
    sum = 0;
    for ( i = 1;  i <= 10;  i += 1 )
        sum += i*i;
    return  sum;
}
```

```
        xor   ecx, ecx
        xor   eax, eax
        inc   ecx
$LL3@cf11:
        mov   edx, ecx
        imul  edx, ecx
        add   eax, edx
        inc   ecx
        cmp   ecx, 10
        jle   $LL3@cf11
        ret
```

编译优化：大小最小化

VC2010集成开发环境

1.1.2 CPU基本功能

- 执行机器指令
- 暂存少量数据
- 访问存储器

1.1.2 计算机基本构成

➢主机五大件

✓中央处理器CPU（主厨）

✓主板（厨房）

✓内存（操作台）

✓硬盘（储物柜）

✓电源（供电、燃气）

1.1.2 CPU基本功能

➢执行机器指令

✓CPU能够直接识别并遵照执行的指令被称为**机器指令**

✓CPU一条接一条地依次执行存放在存储器中的机器指令

✓每一条机器指令的功能通常很有限

✓CPU能够执行的全部机器指令，被称为该**CPU的指令集**

✓CPU决定机器指令。不同种类CPU，其指令集往往不相同

✓按指令的功能来划分，通常机器指令可分为以下几大类：

数据传送指令、算术逻辑运算指令、转移指令

处理器控制指令、其他指令等

1.1.2 CPU基本功能

➢暂存少量数据

✓一个目标程序中的绝大部分指令是对数据进行各种运算或者处理

✓CPU有若干个寄存器，可以用于存放运算数据和运算结果

✓利用寄存器存放运算数据和运算结果，效率最高

✓指令集中大部分指令的操作数据至少有一个在寄存器中

✓CPU内可以用于存放运算数据和运算结果的寄存器数量很有限

1.1.2 CPU基本功能

➢访问存储器

✓CPU要执行目标程序，就要访问存储器。目标程序在存储器中，待处理的数据也在存储器中。这里存储器是指CPU能够直接访问的计算机系统的物理内存

✓存储器（内存）由一系列存储单元线性地组成，**最基本的存储单元为一个字节**。为了标识和存取每一个存储单元，给每一个存储单元规定一个编号，也就是**存储单元地址**

✓CPU支持以多种形式表示存储单元的地址。一些功能较强的CPU还支持以多种方式组织管理存储器

1.1.2 CPU基本功能

➢访问存储器

```
int  x = 1;
int  y = 2;

void  cf12( void )
{
   y = x * x + 3;
   return;
}
```

```
mov   eax, varx3HA
imul  eax, eax
add   eax, 3
mov   vary3HA, eax
ret
```

编译优化：大小最小化

VC2010集成开发环境

1.2 汇编语言

1.2.1 机器指令

1.2.2 汇编格式指令

1.2.3 汇编语言

1.2.1 机器指令

✓把CPU能够直接识别并遵照执行的指令称为机器指令

✓机器指令一般由操作码和操作数两部分构成

✓**操作码**指出要进行的操作或运算

例如，加、减、传送等

✓**操作数**指出参与操作或运算的对象，也指出操作或运算结果存放的位置

例如，寄存器、存储单元和数据等

1.2.1 机器指令

✓机器指令采用二进制编码表示

xor ecx, ecx	33 C9
xor eax, eax	33 C0
inc ecx	41
lab1: mov edx, ecx	8B D1
imul edx, ecx	0F AF D1
add eax, edx	03 C2
inc ecx	41
cmp ecx, 10	83 F9 0A
jle lab1	7E F3
ret	C3

0000 0011 11 000 010

1.2.2 汇编格式指令

✓人们采用便于记忆、并能描述指令功能的符号来表示指令的操作码。这些符号被称为**指令助记符**

✓用符号表示操作数，如寄存器、存储单元地址等

✓由指令助记符、操作符号和常量等表示的指令被称为**汇编格式指令**

1.2.2 汇编格式指令

✓ 汇编格式指令的一般格式

[标号:] 指令助记符 [操作数表]

可省

指令决定操作数的个数

1.2.3 汇编语言

- 汇编语言
- 汇编和汇编程序
- 汇编语言的优缺点

1.2.3 汇编语言

➢汇编语言

- 自然语言是思维的载体，是人与人之间交流的工具
- 程序设计语言是人与计算机之间交流的工具
- 程序设计语言由语句和使用语句的规则组成

✓**汇编语言**是一种程序设计语言，是机器语言的符号化

✓汇编语言的语句主要是汇编格式指令和伪指令

✓由于汇编语言的主体是汇编格式指令，而汇编格式指令又与机器密切相关，且功能有限，所以**常把汇编语言称为低级语言**

1.2.3 汇编语言

➢汇编和汇编程序

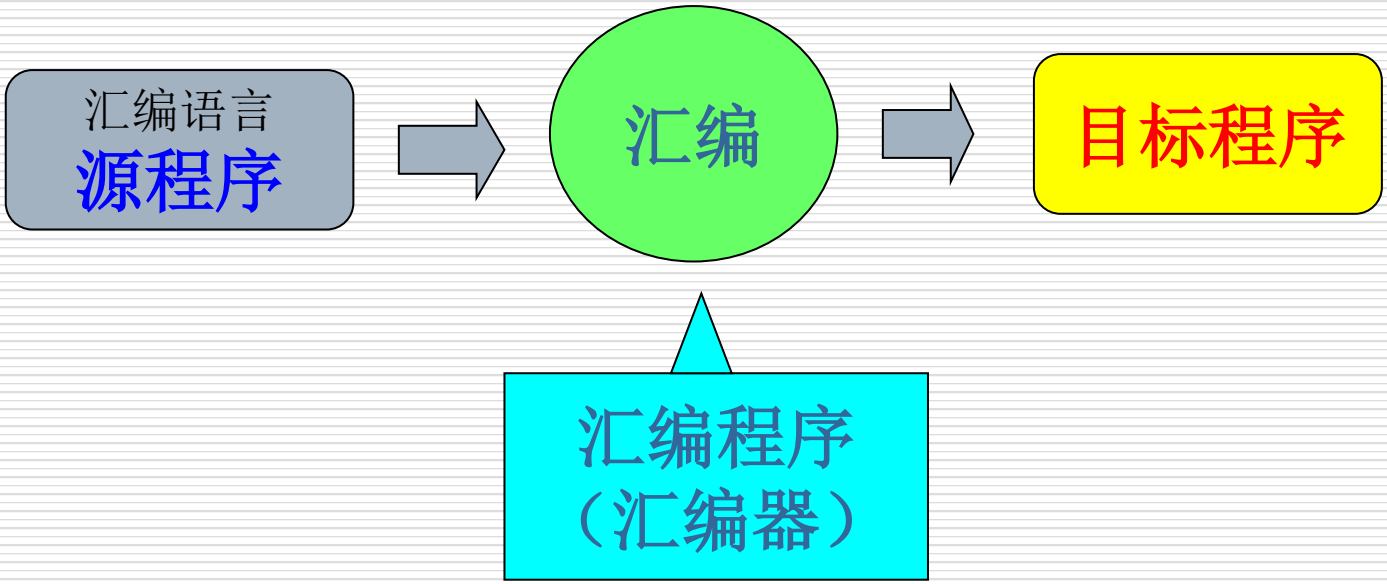
✓把用汇编语言编写的程序称为**汇编语言源程序**，或称为**汇编源程序**，或简称为源程序。

✓把汇编源程序翻译成目标程序的过程称为**汇编**。

✓把完成汇编工作的工具或程序叫做**汇编程序（汇编器）**。

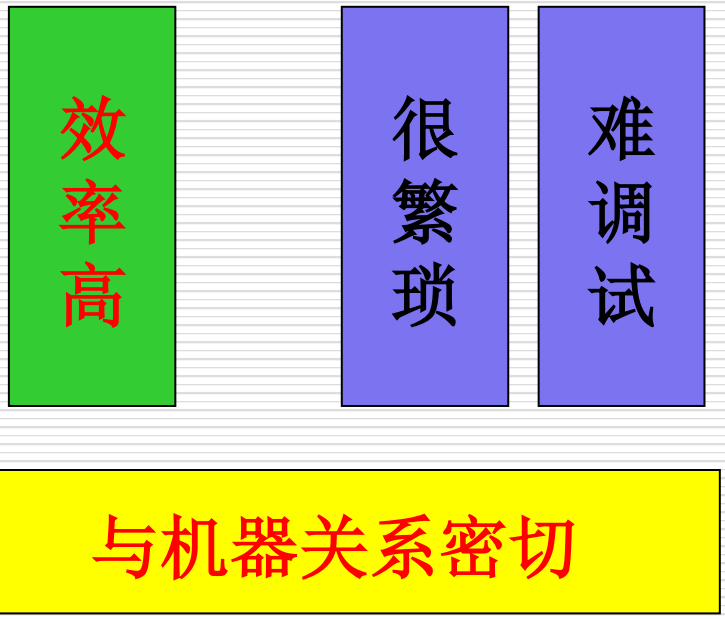
1.2.3 汇编语言

➢汇编和汇编程序



1.2.3 汇编语言

- 汇编语言的优缺点
  - ✓汇编语言与机器关系密切
  - ✓汇编语言程序效率高
  - ✓编写汇编语言源程序繁琐
  - ✓汇编语言程序调试困难



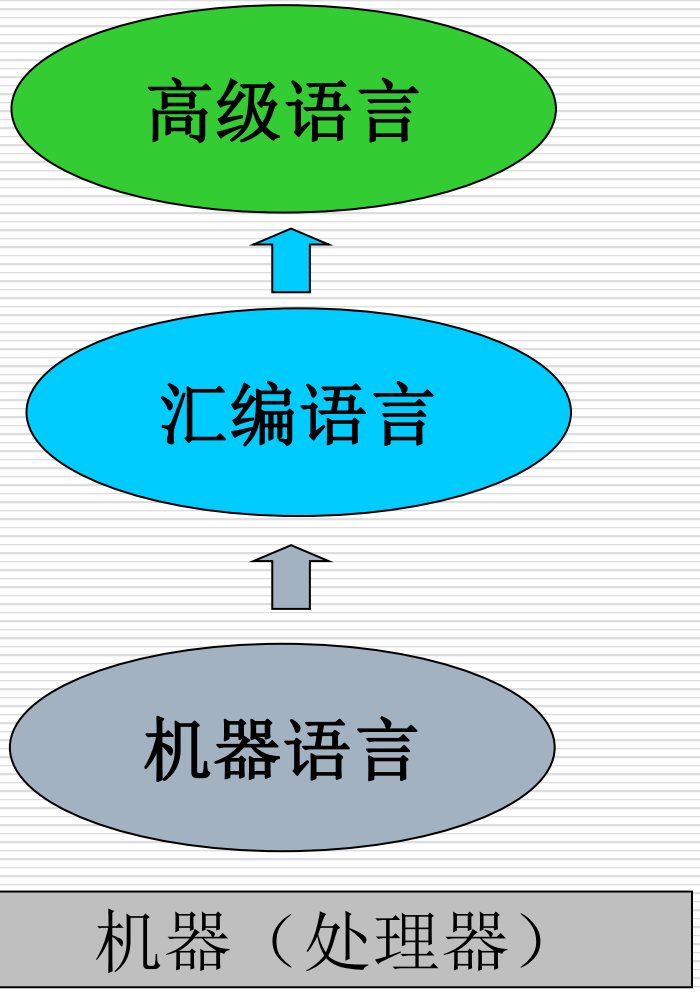
1.2.3 汇编语言

➢语言的发展

```
count = i+3;
```

```
mov   ax,[2000h]
add   ax,3
mov   [2002h],ax
```

```
a10020
050300
a30220
```



*应用汇编语言的场合

- 执行时间/存储容量有较高要求
- 需要提高大型软件效率
- 软件要直接和有效控制硬件
- 没有合适的高级语言

适度地追求“时空”效率!

1.3 数据的表示和存储

1.3.1 数值数据的表示

1.3.2 非数值数据的表示

1.3.3 基本数据类型

1.3.4 数据的存储

1.3.1 数值数据的表示

- 数的二进制表示
- 有符号数的补码表示
- 符号扩展
- 数值数据的表示范围
- BCD码
- 十六进制表示

1.3.1 数值数据的表示

- 数的二进制表示
- 有符号数的补码表示
- 符号扩展

十进制数 21：

8位		00010101	15H
16位	00000000	00010101	0015H
32位	00000000 00000000	00000000 00010101	0000 0015H

H表示十六进制

1.3.1 数值数据的表示

- 数的二进制表示
- 有符号数的补码表示
- 符号扩展

十进制数 **-3**：

8位	11111101	FDH
16位	11111111 11111101	FFFDH
32位	11111111 11111111 11111111 11111101	FFFF FFFDH

1.3.1 数值数据的表示

➢数值数据的表示范围

二进制位数	无符号数	有符号数
8	0 -- 255	-128 -- +127
16	0 -- 65535	-32768 -- +32767
32	0 -- 4294967295	-2147483648 -- +2147483647

1.3.1 数值数据的表示

➢BCD码

十进制数字	8421BCD码	十进制数字	8421BCD码
0	0000	5	0101
1	0001	6	0110
2	0010	7	0111
3	0011	8	1000
4	0100	9	1001

1.3.2 非数值数据的表示

➢西文字符的表示

✓ASCII码

ASCII码是美国信息交换标准码(American Standard Code for Information Interchange)的简称，是国际上比较通用的字符二进制编码。

ASCII表

(American Standard Code for Information Interchange 美国标准信息交换代码)

高四位 / 低四位		ASCII控制字符 0000						ASCII控制字符 0001						ASCII打印字符 0010		ASCII打印字符 0011		ASCII打印字符 0100		ASCII打印字符 0101		ASCII打印字符 0100		ASCII打印字符 0111		
		0						1						2		3		4		5		6		7		
		十进制	字符	Ctrl	代码	转义字符	字符解释	十进制	字符	Ctrl	代码	转义字符	字符解释	十进制	字符	十进制	字符	十进制	字符	十进制	字符	十进制	字符	十进制	字符	Ctrl
0000	0	0		^@	NUL	\0	空字符	16	▶	^P	DLE		数据链路转义	32		48	0	64	@	80	P	96	`	112	p	
0001	1	1	☺	^A	SOH		标题开始	17	◀	^Q	DC1		设备控制 1	33	!	49	1	65	A	81	Q	97	a	113	q	
0010	2	2	☻	^B	STX		正文开始	18	↕	^R	DC2		设备控制 2	34	"	50	2	66	B	82	R	98	b	114	r	
0011	3	3	♥	^C	ETX		正文结束	19	‼	^S	DC3		设备控制 3	35	#	51	3	67	C	83	S	99	c	115	s	
0100	4	4	♦	^D	EOT		传输结束	20	¶	^T	DC4		设备控制 4	36	$	52	4	68	D	84	T	100	d	116	t	
0101	5	5	♣	^E	ENQ		查询	21	§	^U	NAK		否定应答	37	%	53	5	69	E	85	U	101	e	117	u	
0110	6	6	♠	^F	ACK		肯定应答	22	▬	^V	SYN		同步空闲	38	&	54	6	70	F	86	V	102	f	118	v	
0111	7	7	•	^G	BEL	\a	响铃	23	↨	^W	ETB		传输块结束	39	'	55	7	71	G	87	W	103	g	119	w	
1000	8	8	◘	^H	BS	\b	退格	24	↑	^X	CAN		取消	40	(	56	8	72	H	88	X	104	h	120	x	
1001	9	9	○	^I	HT	\t	横向制表	25	↓	^Y	EM		介质结束	41	)	57	9	73	I	89	Y	105	i	121	y	
1010	A	10	◙	^J	LF	\n	换行	26	→	^Z	SUB		替代	42	*	58	:	74	J	90	Z	106	j	122	z	
1011	B	11	♂	^K	VT	\v	纵向制表	27	←	^[	ESC	\e	溢出	43	+	59	;	75	K	91	[	107	k	123	{	
1100	C	12	♀	^L	FF	\f	换页	28	∟	^\	FS		文件分隔符	44	,	60	<	76	L	92	\	108	l	124	|	
1101	D	13	♪	^M	CR	\r	回车	29	↔	^]	GS		组分隔符	45	-	61	=	77	M	93	]	109	m	125	}	
1110	E	14	♫	^N	SO		移出	30	▲	^^	RS		记录分隔符	46	.	62	>	78	N	94	^	110	n	126	~	
1111	F	15	☼	^O	SI		移入	31	▼	^-	US		单元分隔符	47	/	63	?	79	O	95	_	111	o	127	⌂	^Backspace 代码：DEL

1.3.2 非数值数据的表示

➢西文字符的表示

✓ASCII码

大小写字母的编码：

A	B	C	……	Y	Z	大写字母
41	42	43	……	59	5A	十六进制代码
a	b	c	……	y	z	小写字母
61	62	63	……	79	7A	十六进制代码

代码依次递增；
大写字母与对应小写字母之间相差**20H**

1.3.2 非数值数据的表示

➢西文字符的表示

✓ASCII码

10个数字的编码：

0	1	2	3	4	5	6	7	8	9	数字符
30	31	32	33	34	35	36	37	38	39	十六进制代码

代码依次递增；
对应数值基础上加**30H**

1.3.2 非数值数据的表示

- 西文字符的表示
  - ✓ **ASCII码**

特殊符号的编码：

空格	回车	换行	退格	响铃	制表	特殊符号
20	0D	0A	08	07	09	十六进制代码

1.3.2 非数值数据的表示

➢汉字的表示

烫烫烫 锟斤拷的由来

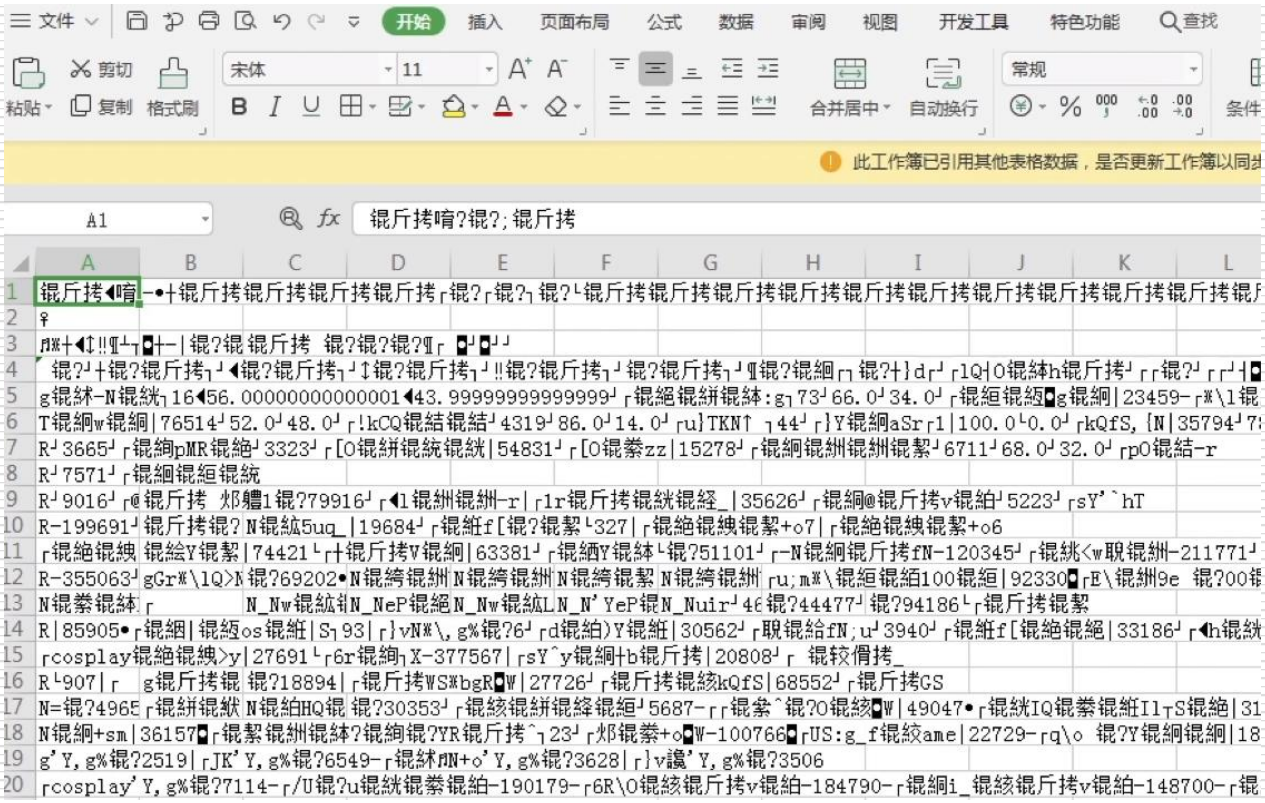
✓变形国标码

- GB2312-80《信息交换用汉字编码字符集——基本集》含有基本汉字符和一般图形字符，共计7445个，其中汉字分成两级共计6763个。
- 国标码是16位编码，高8位表示汉字符的区号，低8位表示汉字符的位号。代码表分成94个区，每个区有94个位。区号和位号都从21H开始。一级汉字安排在30H区至57H区，二级汉字安排在58H区至77H区。
- 变形国标码是16位编码。常用变形方法是把国标码的第15位和第7位均置成1。这种变形方法是在国标码上加8080H。

1.3.2 非数值数据的表示

- 汉字的表示
  - ✓变形国标码

烫烫烫 锟斤拷的由来



1.3.2 非数值数据的表示

➢汉字的表示

✓变形国标码

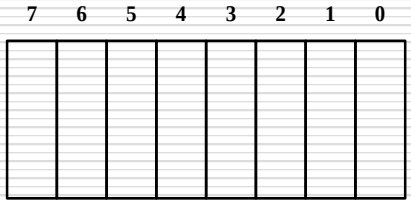
- 排第一的“啊”，编码是B0A1 H
- 紧随其后“阿”，编码是B0A2 H

1.3.3 基本数据类型

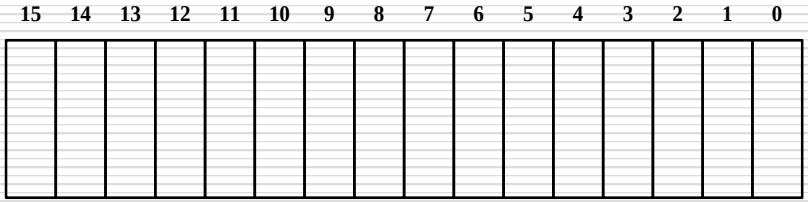
- ✓字节
- ✓字
- ✓双字
- ✓四字
- ✓十字节
- ✓字符串

1.3.3 基本数据类型

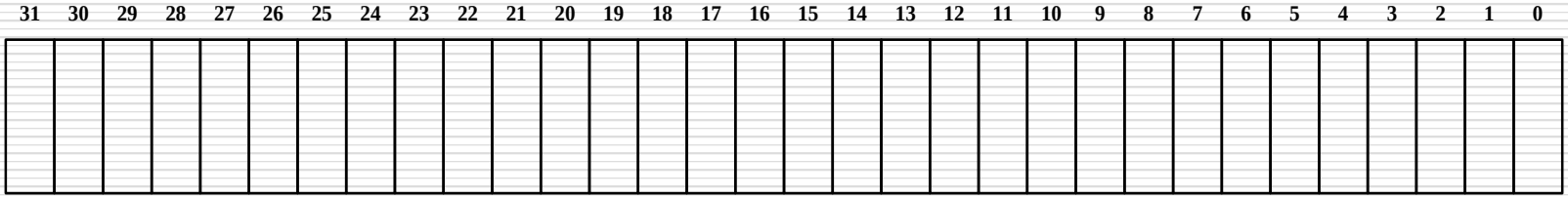
- ✓字节
- ✓字
- ✓双字



(a) 一个字节8个位



(b) 一个字16个位



(c) 一个双字32个位

1.3.4 数据的存储

✓以二进制形式表示的数据和代码存放在存储器（内存）之中。

✓内存由一系列基本存储单元线性地组成，每一个基本存储单元有一个唯一的地址。通常，基本存储单元由8个连续的位构成，可用于存储一个字节的数据。所以，基本存储单元也被称为**字节存储单元**。

✓可以把内存看作为一个很大的一维字符数组，把地址看作为标识数组元素的下标。

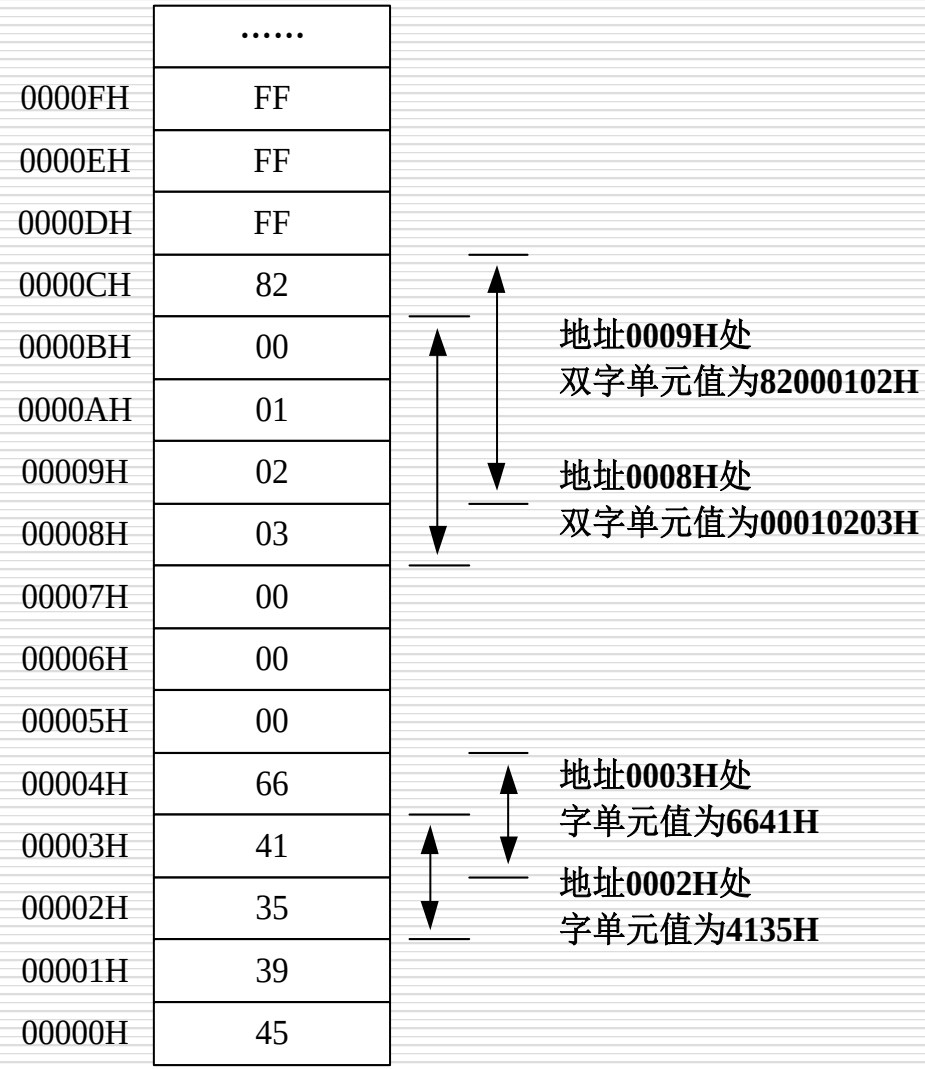
1.3.4 数据的存储

✓字节存储单元是基本的存储单元。

✓每一个字节存储单元中的8位数据的意义，根据需要可以有不同的解释。

✓ **两个**连续的**字节**存储单元构成**一个字**存储单元。字存储单元的地址是较低的字节存储单元的地址。“高高低低”规则。

✓**四个**连续的**字节**存储单元构成**一个双字**存储单元。



1.3.4 数据的存储

```
#include  <stdio.h>
  char  buff[] = { 3, 2, 1, 0, 130 };        // 130=0x82
  int  a, b;
  int  main( )
  {
     char  *p = buff;

     a = *( int* ) p;                          //L1
     b = *( int* ) ( p+1 );                    //L2

     printf( "a = %x, b=%x\n", a, b );     //L3
     printf( "a = %d, b=%d\n", a, b );     //L4
     return  0;
  }
```

a=10203, b=82000102
a=66051, b=-2113928958